Semestrálna práca MI-PAP 2013/2014:

Násobenie matíc

Pavol Kušlita

Martin Klepáč

March 21, 2014

1 Definícia problému

Našou úlohou bolo vytvoriť program, ktorý implementuje násobenie matíc formou ako klasického algoritmu, tak aj s použitím Strassenovho algoritmu.

2 Formát vstupu, výstupu

Formálne, vstup vyjadríme pomocou

- *A, B* = vstupné matice
- *Ax, Ay, Bx, By* = dimenzie vstupných matíc A, B

Výstupom algoritmu je matica *C* s dimenziami *Ax*, *By* za podmienky, že *Ay* = *Bx*. V opačnom prípade nie je možno vykonať násobenie vstupných matíc.

3 Implementácia sekvenčného riešenia

Sekvenčný algoritmu pozostáva z trojice cyklov, ktoré prechádzajú matice *A*, *B* v nasledovnom poradí:

1. riadok matice *A*
2. stĺpec matice *B*
3. stĺpec matice *A*, ktorý zároveň predstavuje riadok matice *B*

Vo najvnútornejšom cykle sa potom vykoná samotný výpočet popísaný pseudokódom nižšie:

```
C[i][j] += A[i][k] * B[k][j];
```

Zložitosť takéhoto výpočtu je potom intuitívne $O(n^3)$. Zároveň tento triviálny algoritmus nevyužíva možnosti cache pamäte.

Vylepšenie tohto triviálneho algoritmu spočíva v použití techniky loop-tiling. Výsledkom je zdvojnásobenie celkového počtu cyklov na 6, pričom miesto sekvenčného posunu indexov *i* až *k* posúvame indexy v našom prípade o 100, aby sme následne iterovali pomocnými indexami *ii*, *jj* respktíve *kk* medzi 0-99, 100-199 a tak ďalej. Výsledkom je rovnaké množstvo vykonanej práce pri efektívnejšom využití cache pamäte. Pseudokód popisujúci techniku loop tiling-u:

```
for i in 0..Ax
 for j in 0..By
  for k in 0..Ay
   for ii in i..i+100
    for jj in j..j+100
     for kk in k..k+100
      C[ii][jj] += A[ii][kk] * B[kk][jj];
```

<TO DO - Strassenov algoritmus sekvencne>

4 Implementácia paralelného riešenia pomocou OpenMP

Prostredie OpenMP sa ukazálo byť veľmi jednoduché na implementáciu triviálneho algoritmu s použitím viacerých vlákien. Snažili sme sa, aby každé vlákno vykonávalo približne rovnako veľkú časť kódu, čo sme dosiahli statickým rozdelením záťaže priamo v prostredí OpenMP. Aby sme v prípade menších vstupných matíc a naopak väčšieho počtu vlákien zamedzili plytvaniu prostriedkov, miesto jedného cyklu paralelizujeme dvojicu vonkajších cyklov s pomocou kľúčového slova *collapse*.

Presný kód definujúci paralelizáciu ako triviálneho algoritmu, tak aj algoritmu s použitím loop tiling-u:

```
#pragma omp parallel for collapse (2) default(shared) private(i,j,k)
schedule(static)
```

<TO DO - Strassenov algoritmus paralelne>

Pre potreby merania na serveri star.fit.cvut.cz sme vygenerovali trojicu pseudonáhodných štvorcových matíc o veľkosti 1024, 2048 a 4096. Program samozrejme dokáže spracovať aj iné ako štvorcové matice, ale pre férové porovnanie Strassenovho algoritmu s ostatnými riešeniami, používame práve štvorcové matice o veľkosti 2^n - naša implementácia Strassena automaticky dopĺňa vstupné matice na túto veľkosť.

Namerané hodnoty sú uvedené v tabuľke 1 pre dané vstupné matice, počet vlákien 1, 2, 4, 6, 8, 12, 24 pre všetky hore popísané implementácie. Výsledné hodnoty predstavujú aritmetický priemer trojice meraní.

	1024x1024			2048x2048			4096x4096		
	CL	**LT**	**ST**	**CL**	**LT**	**ST**	**CL**	**LT**	**ST**
-n 1	11.33	10.02		125.20	79.60		1347.2	651.67	
-n 2	5.84	5.47		56.44	41.22		565.86	327.14	
-n 4	2.93	2.82		30.03	20.79		302.63	163.87	
-n 6	1.96	1.94		20.59	13.96		203.99	109.19	
-n 8	1.48	1.46		16.33	10.57		158.66	81.65	
-n 12	0.98	1.07		10.74	7.05		114.60	56.16	
-n 24	0.92	0.81		8.40	5.87		83.37	50.18	

Table 1: OpenMP meranie [s] (CL = classic, LT = loop-tiling, ST = Strassen)